Removable Security Device

Product Guidelines

Smart Card Reader, USB Security Token, PC USB SIM Card, USB Integrated Chip Card (UICC)

December 2002

Revision 1.10

Intel Confidential

Contents

Figures

Tables

Revision History

Rev.	Description	Date
0.9	Initial release	July 2002
1.00	Add summary guidelines list	August 2002
1.10	Updated to reflect changes from Plugfest for Removable Security Devices	December 2002

1. Introduction

This document is provided as a set of recommendations to offer removable security device (smart card, USB security token, PC SIM card, USB Integrated Chip Card) functions for future mobile PC platforms based on Intel's mobile processors and chipsets.

Platform security is critical to ensure the trust and confidence users have for keeping their own as well as sensitive third party data safe both within the computer as well as when it may be transmitted across a communications channel to another user. Safer computing is a necessity in moving the computer industry forward and is complimentary to the pillars of mobile next generation notebook PCs defined by Intel. The need to protect data and secrets is prevalent in wired communications on corporate networks and Internet traffic, and amplified in a wireless communications environment where data is prone to additional threats. Furthermore, as the mobility of the computer platform increases, the notebook becomes more and more susceptible to theft.

Notebook PCs will progressively build on various security primitives to ultimately deliver "Trusted Client" capability for safe computing. The following sections within this document provide information about the vision for this safer computing environment with focus on the use of removable security devices.

The intended audience of this document includes technical implementers and decision makers within hardware manufacturers of removable security devices used in a notebook PC, notebook PC OEMs, as well as corporate IT planners and implementers intending to use such removable security devices.

1.1. Platform Security Vision

In a notebook platform, a foundation is necessary to provide secure connections for a variety of usage models. Not only must there be a method to authenticate the user for system and network logon, but also to authenticate the platform for network access. User and platform authentication relies heavily on a mechanism of the PC to securely store certificates and keys required to authenticate and ultimately protect connections and transactions.

In a roaming environment where a notebook PC may move from one connection mechanism to another, it is necessary to know the identity of the machine or person requesting access and service. Similarly, in the corporate environment it is desirable to know the identity of the machine, as well as the identity of the user requesting access and service. In both cases, authentication must occur to protect the network from unauthorized use or access. Therefore, both platform identification and user identification are desirable. Each identification mechanism provides useful information for the network provider. Platform identification allows the network to recognize the owner of the platform and the software running on it. User identification allows the network provider to discover who is requesting access. Combined, these two devices provide an enhanced level of security and authentication. Individually, each may provide enough information for a network provider to grant access to the network and certain capabilities.

For a robust solution, each of these identification mechanisms needs a hardware security device to provide the identification and the security functions needed.

The mobile platform security focus using removable security devices is intended to improve the integrity of trusted notebook PCs being marketed today as well as in the future. For those notebook platforms that are unable to incorporate the use of a fixed, motherboard security device (Trusted Platform Module or TPM), removable security devices can provide enhanced platform trust along with focused user authentication. Usage models can be supplemented with the use of a removable security device. Removable security devices can also enhance platform security used with systems that do have a fixed, motherboard security device by providing multi-factor authentication and storage of user secrets. This strategy provides the following benefits:

- Enhance existing security interfaces (e.g. Microsoft* CAPI or PKCS#11) through the hardware key generation and storage capabilities of the security device
- Potentially enhance the security features provided by user authentication, platform authentication, encrypted file system, wireless and wired connections, VPN, and other computer functions
- Provide multi-factor authentication mechanisms

2. Hardware Recommendations

Third Party Vendors (TPVs) of removable security devices should provide hardware that is capable of providing enhanced security through available functions for nonvolatile secure storage, hashing and encryption engines, and random number generation. Functions provided through hardware devices give increased security, as encryption keys never leave the device and algorithms are performed within the device where likelihood of "break in" is extremely small in the daily use of the notebook PC.

2.1. Removable Security Device Types

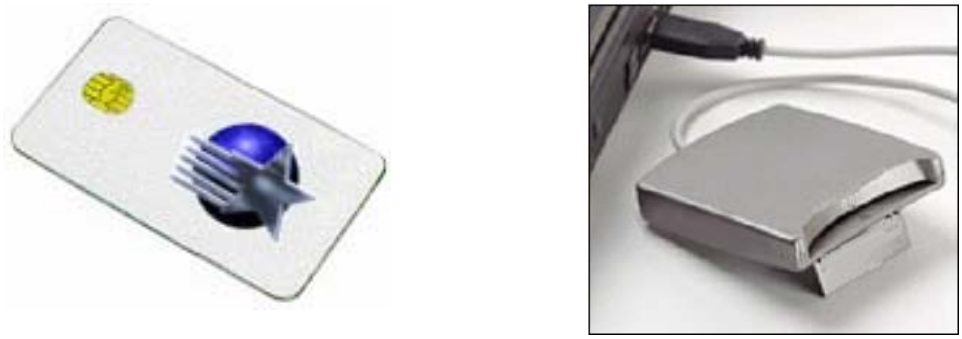
There are a variety of removable security devices that fill the needed capabilities to create a trusted notebook platform. All these device types have similar functionality. The device types addressed by this document include:

- Smart card and its associated smart card reader
- USB security token
- PC SIM card and its associated SIM card reader
- USB Integrated Chip Card (UICC) and its associated reader

2.1.1. Smart Card

Smart cards have been used for years as stored value cards in Europe, but are now being introduced for data security on the Internet and within a PC platform. A smart card is a plastic card the size of a credit card that has an embedded microprocessor for storing and processing information, used for banking, medical alerts, and other transactions requiring cryptographic and secure storage. The smart card is used by inserting it into a reading device that is connected with a notebook computer. The smart card reader can be connected to the notebook computer via USB or PCMCIA.

Figure 1. Example Smart Card and Example Smart Card Reader (USB)



Smart cards used with notebook PCs are referred to as "contact" smart cards. A contact smart card requires insertion into a smart card reader with a direct, physical connection to a conductive micromodule on the surface of the card that is typically gold plated. It is via these physical contact points, that transmission of commands, data, and card status takes place to and from the smart card and the smart card reader connected to the notebook PC.

The chips used in smart cards fall into two categories as well: microprocessor chips and memory chips. A memory chip can be viewed as a tiny storage device with optional built-in security. Memory cards

can hold from 103 bytes to 32k bytes (or more) of data. They are less expensive than microprocessor cards but with a corresponding decrease in data management security. They depend on the security of the card reader for their processing and are suited for use when circumstances permit use of cards with low to medium security.

A microprocessor chip in the smart card can add, delete, and otherwise manipulate information in its own memory. It can be viewed as a miniature computer with an input/output port, operating system, and storage. These microprocessor cards can provide encryption where data can be encrypted and decrypted by the onboard microprocessor driven algorithms. Microprocessor chips are available 8-bit, 16-bit, and 32-bit architectures. Their data storage capacity ranges from 300 bytes to 32k bytes (or more).

The type of smart card used depends on the functions required for a particular application and/or usage model. In order to provide a complete and robust security device used in broad usage scenarios with a notebook PC, Intel strongly recommends that microprocessor smart cards be used rather than "memory only cards".

Additional information for smart cards and smart card readers is available from the Smart Card Industry Association (http://www.scia.org) and from the Smart Card Alliance organization (http://www.smartcardalliance.org).

2.1.2. USB Security Token

A USB security token is a portable, stand-alone device that provides similar functions and features contained in a smart card, just in a different form factor. This class of removable security device connects to the USB port of a notebook PC. It's overall size is approximately the size of a car key; 15 mm x 50 mm x 10 mm. These devices are generally water resistant and come in a tamper evident casing.

Figure 2. Example USB Security Token (Shown With House Key for Size Comparison)

This class of device provides non-volatile secure storage, onboard cryptographic processing, RSA key operations, random number generation, and key generation. Storage on these devices generally ranges from 8 kbytes to 32 kbytes.

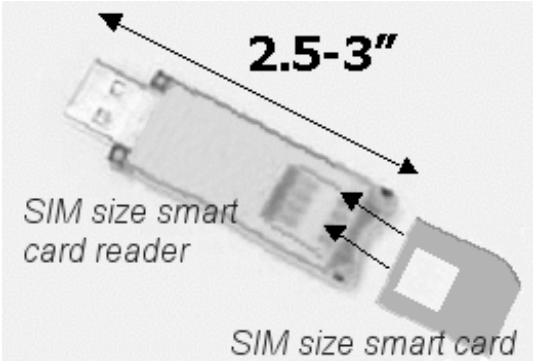
It is not clear that there is a standards body governing this class of device.

2.1.3. PC SIM Card

PC SIM cards and their associated readers offer the flexibility and features provided by both smart card and USB security token solutions in a small, relatively easy to use form factor. PC SIM card readers

take advantage of standard SIM (Subscriber Identity Module) card definitions and available products packaged with a small reader that plugs into the USB port or PCMCIA slot of a notebook PC. The overall size of the USB reader device is similar to the USB security token (15 mm x 50 mm x 10 mm) with the ability to insert and remove the SIM card that contains the security functions of the device. The PCMCIA versions fit the standard size and shape requirements for Type II cards. SIM hardware of the card itself consists of a microprocessor, ROM, persistent EEPROM memory, volatile RAM, and a serial I/O interface. SIM software running on the card itself usually consists of an operating system, file system, and application programs. As with all smart cards, the SIM relies on the card terminal – in this case, the USB reader device – for power and clock.

Figure 3. SIM Card and Example USB SIM Card Reader



SIM cards in their small form factor are created to be compliant with an industry standard definition GSM 11.11 available through the SIM Alliance (http://www.simalliance.com/). SIM cards can provide security features including: 40-bit encryption key; GSM-defined secret algorithms, non-volatile storage for keys, certificates, phone numbers, network profiles, and other personal information; and other functions similar in nature to those provided by smart cards.

These devices create another new form factor with functions and features similar to the others already described. The SIM card that is inserted into the small reader, although smaller in size, is still compliant with other smart card industry standards.

2.1.4. USB Integrated Chip Card (UICC)

USB Integrated Chip Card (UICC) offers the same functionality as a smart card with its associated reader with added flexibility. This reader for this type of device is a passive "pass through" for the USB connection. All USB functions are bundled onto the card along with the smart card functions. When the reader is plugged into a PC, it is not detected because that logic resides on the card itself. When the card is inserted into the reader, the PC recognizes the device.

UICC cards are the same size and form factor as SIM cards, but can also be found in full size smart card form factor. The reader can be in a small form factor similar to the USB security token to accommodate the small SIM card or the reader can be a full size form factor to accommodate the full size smart card.

Figure 4 below represents the USB Integrated Chip Card (UICC). The outlined area of the card can be manufactured with perforations so the SIM form factor can be punched out and used in a different reader.

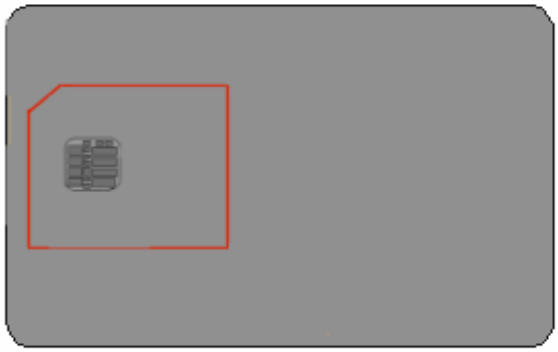


Figure 4: USB Integrated Chip Card (UICC)

2.2. Mechanical / Attach

Mechanical / electrical attachment of removable security devices to a notebook PC is recommended through either USB ports or PCMCIA slots. Other methods of connecting to a notebook PC are not recommended.

USB ports are found on most, if not all, notebook PCs delivered to the marketplace today. USB definitions provide for adequate functionality as well as power management necessary to promote a good user experience when used with a notebook PC. Information related to USB is available from the USB Implementers Forum*, Inc, a non-profit corporation founded by the group of companies that developed the Universal Serial Bus specification. (http://www.usb.org).

PCMCIA (Personal Computer Memory Card International Association) or PCCard slots are found in all notebooks and provide another convenient method for connecting a removable security device to the notebook. PCMCIA information is available from the PCMCIA forum web site (http://www.pcmcia.org).

2.3. Power Management

Because these devices are intended for use in a notebook PC where battery power should be maximized to improve the user experience, vendors should provide removable security devices designed, developed, and tested for low power consumption.

All devices intended for use in a mobile environment where the PC spends any amount of time running from battery power should be optimized for minimum power consumption. Devices and their controlling software should take steps to prudently use power when operating and take further steps to reduce power consumption when idle. The ACPI specification gives guidelines for device characteristics when idle.

The goal of power management is to conserve power while the computer and its devices are working and to put the computer to sleep when it is not working. The decisions that determine how to save energy and when to go to sleep are implemented by the *power policy*. The implementation of power policy is distributed throughout the system, with different system components acting as *policy owners* for different aspects. For example, the operating system is the policy owner that is responsible for determining when the computer should go to sleep, determining the level of sleep into which the computer should go, and knowing how to operate the processor to reduce power consumption, heat, and noise.

Carrying out power policy — actually controlling devices so that power consumption or capabilities change — is the responsibility of the device drivers for the affected device and is shared among the drivers in the stack. Device-specific software drivers, known as *minidrivers* for Windows* operating systems, are responsible for saving and restoring the device's settings across transitions to and from low-power states. When a policy owner makes the decision to put a device into a low-power state and communicates that to the device driver through the system, the device driver saves the device's settings and sends the request to the bus driver. The bus driver gives the command to the device to enter the low-power state. When the device is turned on, the bus driver powers up the device, and the device-specific driver restores the saved context.

System power states derive from the Advanced Configuration and Power Interface (ACPI) specification. They are defined as follows:

- S0/Working—The CPU is fully up and running; devices are powering up and down as needed.
- S1—The CPU is stopped; RAM is refreshed; the system is running in a low-power mode.
- S2—The CPU has no power; RAM is refreshed; the system is in a lower power mode than S1.
- S3—The CPU has no power; RAM is in slow refresh; the power supply is generally is a reduced power mode (for example, it can't supply much power and is running in a lower power efficiency mode).
- S4—The hardware is completely turned off; system memory has been saved to disk.
- S5/Off—The hardware is completely off; the operating system has shut down; nothing has been saved.

System power states are primarily defined in terms of motherboard/CPU/core logic characteristics and don't relate to individual devices themselves.

Every device in the system has its own power states. These are defined as follows:

- D0—Fully on.
- D1—Low power; context may be saved in hardware (depends on the device class).
- D2—Low power; context may be lost by hardware (depends on the device class).
- D3—Power may be lost; all context lost by hardware.

Devices that are not being used should be put into low-power states to conserve power. The responsibility for power-managing devices is divided into three parts:

- Policy owners decide when a device should switch power states.
- Device drivers save and restore any hardware context lost by switching to a low-power state.
- Bus drivers (including the ACPI System Bus driver) use standard interfaces to signal hardware to switch to a low-power state; they may possibly remove power from the device.

Processor power states (C*x* states) are processor power consumption and thermal management states within the global working state of the notebook system. Power conservation of the notebook can be improved as the processor is allowed to transition to lower power states as defined below. Removable security devices plugged into the USB port can have an impact on the ability for the processor to enter C3 as described in Section 2.3.2 of this document.

- *C0* -- While the processor is in this state, it executes instructions.
- *C1* -- This processor power state has the lowest latency. The hardware latency in this state must be low enough that the operating software does not consider the latency aspect of the state when deciding whether to use it. Aside from putting the processor in a non-executing power state, this state has no other software-visible effects.

- *C2* -- The C2 state offers improved power savings over the C1 state. The worst-case hardware latency for this state is provided via the ACPI system firmware and the operating software can use this information to determine when the C1 state should be used instead of the C2 state. Aside from putting the processor in a non-executing power state, this state has no other software-visible effects.
- *C3* -- The C3 state offers improved power savings over the C1 and C2 states. The worst-case hardware latency for this state is provided via the ACPI system firmware and the operating software can use this information to determine when the C2 state should be used instead of the C3 state. While in the C3 state, the processor's caches maintain state but ignore any snoops. The operating software is responsible for ensuring that the caches maintain coherency.

For further details on optimizations and compliance for power friendly software device drivers and hardware devices, refer to the OnNow white papers, presentations, and device class power management reference specifications: www.microsoft.com/hwdev/onnow/default.htm. Additional information for power management of devices and PC platforms is available from the ACPI organization at http://www.acpi.info.

2.3.1. Device Power State Control

Software device driver support for each device should correctly handle power state transitions of the device itself as well as power state transitions within the PC system. Vendors should use the following checklist while implementing power management in software device drivers:

- Add support for setting power capabilities.
- Add support to handle device power state transitions.
- Add support to handle system power state transitions.
- Add support for handling application interfaces.

2.3.1.1. Support for ACPI Power States

In general terms, devices and the device drivers that control them should be optimized for power efficiency to ensure good battery life in the notebook PC. The ACPI specification defines how devices should behave during system power state transitions as well as device power state capabilities. Refer to the ACPI specification for complete details (http://www.acpi.info).

Power management features within Windows*, especially standby and hibernation and resuming from those states, rely on WDM device drivers that support power management in addition to compatible hardware.

Different policy owners manage system power states and device power states separately in the system. In general, device drivers do not need to concern themselves with system power states except to save and restore as the system suspends or resumes. However, the two types of states do interact and some understanding of this is necessary for device driver developers.

In the low-power system states S1-S3, the system cannot support having all devices On. System designers make tradeoffs in designing their motherboards, based on how much power they want to consume in each state and, therefore, which devices will be powered in each state.

This device state capabilities matrix implies a maximum device power state that the device can be in for any given system state. Device drivers need to use this information in deciding how to appropriately manage their device when the operating system requests to put the system to sleep. In most cases, the device state for any system sleep state is D3, but this may be affected if wakeup is enabled.

2.3.1.1.1. ACPI S-states

From the device driver's perspective, the operating system puts the computer into a sleeping state (S-state) using the following steps:

1. Query phase, during which the operating system asks each driver if it can go to sleep now by sending it a QUERY_POWER IRP with a system power state.
2. Sleep phase, during which the operating system tells each driver to go into a low-power state by sending it a SET_POWER IRP with a system power state.
3. Signal hardware phase, during which the ACPI driver signals the motherboard chip set to go to sleep.

Query Phase

The operating system determines whether the machine can be put into a system sleeping state and which system sleeping state to use by sending *system* power state queries to each device driver. The operating system begins by sending queries for the lowest power system sleeping state the platform supports. If a driver fails a query, the operating system sends the queries again, this time with the next higher power system state and repeats the process until all drivers succeed or the operating system runs out of power states.

1. The operating system sends each driver (in sequence from leaf nodes towards the root) a IRP_MJ_POWER/IRP_MN_QUERY_POWER IRP with a requested system state.
2. Upon receiving the QUERY_POWER IRP, the device driver determines whether it can go into that system sleep state and returns success or failure accordingly. Device drivers should fail this query only for the following reasons:
    a. Activity is taking place on the device that would cause end-user data loss if the system were to enter a sleeping state at this time.
    b. The device is enabled for wake up and according to the SystemWake field of the DEVICE_CAPABILITIES structure, the device cannot wake the system in this power state. Removable security devices do not fall into this category.
3. If any device driver returns success, it must stop any activity on the device that would prevent the system from going to sleep (that is, any activity that would cause the driver to report failure to this message).
4. If any device driver fails the query, the operating system attempts higher system power states until it finds a state that will work or it runs out of states. If the operating system runs out of power states, it sends a SET_POWER IRP with a system state of SystemWorking to indicate that the system will not be going to sleep and drivers can resume normal activity.

Sleep Phase

During the Sleep phase, the operating system tells each device driver to go into the selected system sleeping state.

1. The operating system sends each driver (in the same sequence used for queries) an IRP_MJ_POWER/IRP_MN_SET_POWER with the targeted system power state.
2. Upon receiving the SET_POWER IRP, the driver determines which device power state to go to as follows:
    a. If the device is not enabled for wake up, the device goes to D3.

b. If the device is enabled for wake up, the device goes to the power state indicated by DeviceState field of the DEVICE_CAPABILITIES structure (that is, it goes into DeviceState[irpStack->Parameters.Power.State.SystemState]).

3. The device driver switches to the chosen *device* state by calling PoRequestPowerIrp. This will cause the process to take place as described in the section "Switching to a Low Power State" earlier.

The device driver completes the SET_POWER IRP it received in step 2.

Signaling System Wake Up

When the system wakes, the device drivers work together to determine the source of the wake up as follows:

1. When a bus driver finds that its bus is signaling wake up (by seeing its WAIT_WAKE IRP complete), it uses the mechanisms defined in the standard for that bus to determine which children are signaling wake up.
2. The bus driver completes the WAIT_WAKE IRP for each device-signaling wake up.
3. In the completion routine, each child device handles the wake up event. This may require turning on the device by calling PoRequestPowerIrp. If the child device driver is a bus driver, it should repeat the process at step 1.

2.3.1.2. Device Power Management in WDM Device Drivers

Device drivers in the Windows operating system follow the WDM (Windows Driver Model) architecture. WDM provides facilities for drivers to implement power policy and control. Device Drive Interfaces (DDIs) are defined for synchronizing power state changes with other power management activities in the system and for detecting device idleness. I/O request packets (IRPs) are defined for setting power state, enabling wake-up, and determining device capabilities.

Power management policy decisions generally result in the kernel sending an IRP to a specific device object (often the policy owner itself) to affect some power management control. The IRP is passed along to the device object's parent as needed to cause the actual control functionality. This can involve many sets of child-parent pairs between the policy owner and the actual power control. Drivers implementing power policy or control use kernel services through the WDM DDIs.

Device drivers determine the supported power states using the following procedure:

1. When the device driver loads, the operating system will send the device's Functional Device Object (FDO, which is owned by the device driver) an IRP_MJ_PNP/IRP_MN_QUERY_CAPABILITIES IRP to determine the capabilities of the device.
2. The device driver must first forward this IRP to its Physical Device Object (PDO, which is owned by the bus driver) to determine what capabilities are supported by the hardware.
3. When the bus driver receives the QUERY_CAPABILITIES IRP, it interrogates the hardware using standards for that bus and fills out the DEVICE_CAPABILITIES structure, including the DeviceWake, SystemWake, and DeviceState fields. These fields are defined as follows:

    a. DeviceState—the highest power device state the platform can maintain for each given system state. This matrix is filled in by only one driver, the ACPI System Bus driver. Device drivers and bus drivers do not have access to the system-specific information required to create this matrix.

b. DeviceWake—the lowest power device state for which the device can signal a wake event. This value is set to PowerDeviceUndefined if the device cannot signal a wake event. Removable security devices such as a smart card reader must implement this function to ensure the reader wakes up when the smart card is inserted or removed. USB security tokens where the "reader is built-in" may not need to implement this function.

c. SystemWake—the lowest power system state in which this device can signal wake. This value is filled in by only one driver, the ACPI System Bus driver. Device drivers and bus drivers do not have access to the system-specific information required to fill in this value. Removable security devices should not support this capability.

4. When this IRP completes, the device driver adjusts these capabilities based on the capabilities of the driver itself, and any private information it has about the device hardware. Note that the device driver can only remove capability; it cannot add capabilities.

5. The device driver completes the IRP to the operating system.

When the policy owner decides to put a device into a low-power state, the following actions occur:

1. The policy owner calls PoRequestPowerIrp to create an IRP_MJ_POWER/IRP_MN_SET_POWER IRP for the required device state and send it to the FDO at the top of the device's IRP stack.

2. When the device driver receives the SET_POWER IRP, it saves any device context it will need to restore the device to the D0 state.

3. The device driver calls PoSetPowerState to tell the operating system that the device is leaving the D0 power state.

4. If necessary, the device driver may exercise any hardware controls it has to put the device into the targeted power state.

5. As with all other POWER IRPs, the device driver must forward the IRP to its PDO by calling PoStartNextPowerIrp (to indicate to the system that it is done with the current IRP and that the next one may be sent), then PoCallDriver (to send the current IRP to the PDO).

6. When the bus driver receives the SET_POWER IRP for its child device, it exercises any hardware standard controls to put the device into the targeted power state. It then completes the IRP.

When the policy owner decides to turn a device on (due to a received I/O request that requires the device to be on), the following actions occur:

1. The policy owner calls PoRequestPowerIrp to create an IRP_MJ_POWER/IRP_MN_SET_POWER IRP for the D0 state and send it to the FDO at the top of the device's IRP stack.

2. When the device driver receives the SET_POWER IRP, it merely fills in a completion routine, then forwards the IRP to its PDO using PoStartNextPowerIrp and PoCallDriver.

3. When the bus driver receives the SET_POWER IRP at the PDO, it exercises its standard hardware mechanisms to restore power to the device and then completes the IRP.

4. In its completion routine, the device driver restores all device context to return the device to D0.

5. The device driver calls PoSetPowerState to tell the operating system that the driver is now fully in the D0 power state.

2.3.2. USB Power Management Support

For removable security devices connected to a USB port, additional care should be taken for power management. Support should be provided for the following:

- **USB Power Management Capabilities Query**. USB device capabilities are reported to the USB Host via the standard Power Descriptors. These address power consumption, latency time, wake support, and battery support and status notification.
- **USB Power Management State Transition Commands**. USB device power states are controlled by the USB Host via the standard SET_FEATURE command. USB device power states are queried via the standard USB GET_STATUS command.

In a notebook computer with USB device ports, the processor may not be able to enter the C3 power savings mode. (See ACPI spec for further details on C3.) This problem can occur because the active bus polling that is required to maintain communication with universal serial bus (USB) devices can prevent the CPU from switching to a C3 power-saving state. By default, the USB polling interval is set for every one millisecond (ms). Even if no devices are connected to the USB controller, the polling operation is still performed and the processor cannot enter a C3 power-saving state. This situation is improved with operating support for device driver operations called *selective suspend*.

Refer to the Universal Serial Bus Implementers Forum (USB-IF) Web site, at http://www.usb.org/ for additional detail.

2.3.2.1. USB Selective Suspend

In Microsoft Windows XP* and later operating systems, the USB core stack supports a feature known as "Selective Suspend." This feature allows a device driver to turn off the USB device it controls when the device becomes idle, even while the system itself remains in a fully operational power state (S0). This feature is primarily intended to conserve battery power in notebook PCs. It should be recognized that removable security devices such as smart cards or security tokens connected to the USB port are used relatively infrequently during the operation of the notebook PC. As such, implementing this support is essential to provide reasonable battery life for the user. The consequence of not implementing this support results in unnecessary battery life reduction.

Implementing this simple mechanism provides power reduction in three instances: one for the USB controller, the USB device itself, and for the host CPU in the notebook PC. The USB controller, in particular, often uses a lot of battery power, even when there are no devices attached to the system. Furthermore, an active USB controller prevents the host processor from transitioning to C3 state, which would provide additional power savings. The Selective Suspend feature allows drivers signal the host controller that it is idle (and able to enter a low power mode), as well as indicate other items are idle including empty USB hubs, the Root Hub, and the full range of USB devices supported by the operating system.

Removable security devices attaching to the notebook PC using the USB port should consider the following:

- Smart card readers can be considered inactive when no card is present in the reader. The device should implement some form of remote wake-up such that when the reader is empty (no card), the reader is idle and Windows is notified using Selective Suspend. When the card is inserted, the reader detects that event and wakes up.
- The removable security device should be considered inactive when no application requests have been made for 1 minute.

Selective Suspend is implemented by means of the **IOCTL_INTERNAL_USB_SUBMIT_IDLE_NOTIFICATION** IOCTL, also known as an "idle request." Idleness for a removable security device is measured in terms of seconds of inactivity (no requests). Idleness detection should be implemented regardless of whether the reader has a smart card

inserted or not. A function driver whose device is idle must send this IOCTL to notify the USB core stack that its device is ready to be turned OFF. The USB Root Hub driver, or the USB Generic Parent driver in the case of a composite device, receives the idle request and holds it pending. However, the hub driver does not actually turn OFF the device. That is done in a callback routine of the device's own function driver. The callback routine is passed down the stack along with the idle request, and either the USB hub driver or the USB Generic Parent driver calls the callback routine once it is safe to turn the device OFF. The idle request IRP remains pending until the hub driver receives a request to turn the device ON.

Once suspended, a removable security device can be awakened in one of two ways:

1. Resumption of signaling occurs for remote wake-capable devices (that is, the Wait/Wake IRP completes successfully). Typically, the device is set to a fully operational power state (D0) by the function driver's Wait/Wake IRP completion routine. For removable security devices, this might be initiated when a smart card reader with a user keypad sends a signal to wake up to process a request or an associated biometric device is activated by the user or simply when a smart card is inserted into the reader.
2. A new open request is initiated by client software. This is the mechanism expected to awaken a removable security device that is in a low power mode. This is probably the most common case for a removable security device.

If the device can be set to a fully operational power state (D0), the hub driver completes the idle request with a status code indicating success. This informs higher-level drivers that the device is no longer idle. If, however, the device is in a non wake-capable power state such as D3, or it has been removed or stopped, the hub driver completes the idle request IRP with an appropriate error code.

A function driver may cancel the idle request after it has been made, but must do so before the hub driver calls the callback routine. If the function driver cancels the idle request IRP, the hub driver will complete the IRP with an error code. When a function driver's completion routine detects that the hub driver completed the idle request IRP with an error code other than STATUS_POWER_STATE_INVALID, it should attempt to turn on the device. This should be done because the hub driver does not guarantee that the device is in a powered state whenever it completes an idle request IRP with an error.

When the system is not in the working state such as in S3, the ACPI bus driver completes all Wait/Wake IRPs; whereas if the system is turned ON, the USB controller driver must handle Wait/Wake IRPs for USB devices. For this reason, support for Selective Suspend in the USB core stack is disabled by default in upgrade installations of the operating system (Windows XP and later). For clean installations, the Selective Suspend feature will be enabled. Selective Suspend is disabled for upgrade installations, because some USB host controller hardware does not support selective suspension of its ports and therefore, cannot indicate a resumption of signaling unless the entire controller is suspended. Regardless of the default state set up either by default or user configuration, device vendors should assume that this feature is enabled and provide appropriate support as defined here.

To enable Selective Suspend support for a given Root Hub and its child devices, a user must select the checkbox under the Power Management tab for the USB Root Hub in the Device Manager. However, function drivers should not try to determine whether selective suspend is enabled before sending idle requests. They should *always* send the requests whenever the device is idle. If Selective Suspend support is not enabled, the hub driver will complete the idle request with an error.

Additional information for Selective Suspend can be referenced in other white papers found on both Intel and Microsoft web sites as well as in the Windows XP DDK.

2.3.3. PCMCIA Power Management Support

PCMCIA devices (both 16-bit and 32-bit Cardbus) do not have device power states defined. The only power states available are ON and OFF, controlled by the host bus controller. The Cardbus specification is a superset of the PCCard-16 specification, incorporating the power management specification for PCI bus. Power management capabilities query, state transition commands and wake event reporting are identical.

Despite not having multiple power states defined, it is expected that removable security devices plugged into the PCMCIA slot prudently manage power. Circuitry within the device not doing work should be autonomously turned OFF or put into a low power idle state. The device must be capable of restoring from its low power state to a working state to handle device requests.

Furthermore, the device and device driver for PCMCIA removable security devices should properly handle system suspend / resume cycles as outlined in the ACPI specification.

Refer to the PCMCIA (Personal Computer Memory Card International Association) Web site at http://www.pc-card.com/.

3. Overview Testing

This section provides simple overview checklist testing for each product.

3.1. Device Hardware Configuration

Removable security devices should connect to the notebook computer either using the PCMCIA slot or by the USB connector. Because serial ports are considered legacy connections that should soon start to disappear from PCs, such connections should not be used for removable security devices.

PCMCIA devices can use either the PCCard-16 or PCCard-32 (Cardbus) interface type. Use of a PCMCIA removable security device must not preclude use of other PCMCIA devices that may be blocked or hindered by the security device protruding too far.

3.2. Software Install, Configuration, and UnInstall

Configuration of these devices should be an automatic process initiated when the device is plugged into the system causing the Windows device installer to search for device driver software for the new removable security device.

If the device is plugged into either USB or PCMCIA, then Windows should search for the proper device driver. If it is not found as part of the default Windows installation, Windows checks media already in the CDROM drive. If not found, the user will be prompted to insert the right media with the device driver for the removable security device. Inserting the CDROM should allow the Windows device installer to find the correct INF file in the root of that media to enable installation of software for the hardware device. The INF file should provide all the information necessary to install all the required support pieces for the removable security device including the device driver, MS-CAPI CSP, PKCS#11, and other vendor specific utilities and configuration. The vendor may alternatively provide software installation through a “setup” utility that installs all necessary software for the device. Although it is preferred to have an INF file in the root directory of the installation media, the user setup software installation is also desirable.

A software installation utility should be provided on the security product's CDROM media provided by the vendor. Inserting the CDROM should automatically start (autorun) the installation program to walk the user through the installation steps. If software installation is done prior to inserting / plugging in the removable security device, then the Windows device installer should properly install and configure the device driver so that the Windows device installer should not request further interaction when the device is plugged in.

The software installation utility should provide a simple and easy to use interface. Complicated configuration options should not be required. Software should install and work properly without requiring the user to reboot the system. Device drivers should be dynamically removed by the OS when the device is unplugged such that the device driver does not show up in the Windows Device Manager unless the device is attached. This means that the device driver and support software should be capable of dynamic installation rather than static installation.

The software installation process should configure a means to uninstall this software through the Control Panel "Add/Remove Software" menu. The uninstall process should cleanly remove all software for the device. Removable security devices that have device driver support bundled with the operating system are not expected to have provisions to uninstall that software support.

A software utility should be included with the product to allow the user to examine and manipulate the contents of the device. This utility may provide a means to change the device password, remove certificates, and/or refresh/format the device.

3.2.1. Installation With Multiple Operating Systems

The removable security devices should be tested under both Windows 2000 and Windows XP operating systems. As each operating system has unique characteristics, it is important to test under both operating environments.

3.3. Cryptographic Software Interface Support

Support for Microsoft CAPI should be either a standard part of the operating system or installed with device drivers for the hardware. The same is true for the PKCS#11 support: it should ship as a standard part of the operating system, or should be installed with hardware device drivers.

Support for CAPI and PKCS#11 apply only to devices that deliver cryptographic functions by themselves. This support is not to be tested for smart card readers, but does apply to USB security tokens or other such devices that provide cryptographic operations.

CAPI CSP Support

Software support should be delivered with the security device that is in compliance with the Microsoft CAPI v2.01 interface with a CSP certified and signed by Microsoft. Verification can be done using the CSPTESTSUITE software utility from Microsoft.

PKCS#11 Support

Because there is no "registration" mechanism defined for PKCS#11, the DLL name providing support for the hardware security device must be known. The PKCS#11 support should be written to the Cryptographic Token Interface Standard v2.01 or later as defined by RSA*.

3.4. WHQL Tests

Microsoft WHQL tests are available for download from the Microsoft Developer web site. It is expected that all removable security device vendors have passed these WHQL tests. No WHQL specific tests should need to be duplicated to accommodate this guideline. However, it is important to check whether the specific product meets WHQL compliance by checking the Microsoft supported products lists.

Background: What do Logo'd Products Offer for End Users?

The "Designed for Windows" logo can play a key role in reducing total cost of ownership for organizations. The Windows logo criteria is one of Microsoft's key vehicles in communicating to software developers how to incorporate Zero Administration for Windows (ZAW) initiative features into their applications in preparation for the next releases of Windows.

Businesses that use products meeting the Designed for Windows logo criteria stand to gain the following benefits:

- **Lower support costs:** Logo'd applications follow standard Windows look-and-feel guidelines. That helps users get up to speed without phoning your helpdesk.
- **Help manage "DLL Hell":** Logo'd applications do not overwrite applications or uninstall key components.
- **Support mixed Windows environments:** Logo'd products work in Windows environments to help manage the mix of Windows desktops in an organization
- **Proper use of operating system:** Logo'd products make proper use of the Windows registry and other key operating system files.
- Compliance with the Americans with Disabilities Act and other equal rights legislation: Logo'd applications meet usability standards for a wide range of people.

The Microsoft WHQL compliance list is updated regularly and is found at:

http://www.microsoft.com/windows/catalog/wcbody.asp?subid=22&catid=6bc37356-d760-4736-9a32-baf3a2c3f34e

3.5. Overview Guidelines Checklist

The following checklist summarizes the important aspects of a removable security device to meet the guidelines defined in this document. All devices should meet these recommendations in each case.

This short list of evaluation points is not intended in any way to be a comprehensive validation of any particular product. It is however intended to represent general "mobile friendliness" of devices as described in this set of vendor recommendations.

Line items that indicate a "NA" for the compliance metric should only be marked if the device does not fall into that category. For example, a PCMCIA device should not attempt to measure compliance against USB recommendations.

Table 1: Overview Guidelines Checklist

Guideline	Functions as Expected (Windows 2000)	Functions as Expected (Windows XP)
1. Device connection via USB or PCMCIA	Yes No	Yes No
2. All software entities that should be installed to a user's hard drive should be described in an INF description file shipped with these collaterals on the root directory of the product software support CDROM. Alternatively, software installed through a "setup" utility.	Yes No	Yes No
3. PCMCIA devices are automatically detected when inserted for the first time, and either automatically configured, or prompt the user to insert media with software support. Software needed for this device is installed with minimal user interaction.	Yes No NA	Yes No NA
4. USB devices are automatically detected when inserted for the first time, and either automatically configured, or prompt the user to insert media with software support. Software needed for this device is installed with minimal user interaction.	Yes No NA	Yes No NA
5. All software collateral should be delivered on CD-ROM. When inserted, the CDROM should "autorun" for installation.	Yes No NA	Yes No NA
6. Device drivers should be fully functional after simple installation *without restarting the PC*. (dynamic device driver rather than static.)	Yes No	Yes No
7. A mechanism for uninstalling the software should be provided after installation so the user can remove software through the Control Panel "Add/Remove Software" menu	Yes No NA	Yes No NA
8. User configuration and installation should be simple (ease of use): includes a user-friendly, simple to understand interface.	Yes No	Yes No
9. All software collateral should be supported in both Microsoft Windows 2000* and Windows XP* operating environments.	Yes No	Yes No

Guideline	Functions as Expected (Windows 2000)	Functions as Expected (Windows XP)
10. All software device drivers for removable security devices should have been tested and signed by Microsoft for use with Windows 2000 and Windows XP. Device drivers should be WHQL certified. Installation should occur without warning by the Windows device installer.	Yes No	Yes No
11. Device found on Microsoft WHQL compliance "good" list? (Check the Microsoft web site for the latest list.)	Yes No	Yes No
12. Device found on Microsoft WHQL "Designed for Windows XP" device list (Check the Microsoft web site for the latest list.)	NA	Yes No
13. Software should include a utility to read configuration information from the removable security device	Yes No	Yes No
14. Support should be provided for the removable security device in compliance with the Microsoft CAPI v2.01 interface with a CSP certified and signed by Microsoft.	Yes No NA	Yes No NA
15. Support should be provided for the hardware security device in compliance with the PKCS #11 interface Cryptographic Token Interface Standard v2.01 or later as defined by RSA*.	Yes No NA	Yes No NA

4. Application Testing

Application testing with run real world applications to ensure the removable security device functions in an expected fashion. Although testing in this section is not exhaustive, it provides a representative sample of real application use of the removable security device.

Smart card readers are expected to perform these tests to ensure the reader provides the pathway to the smart card itself. Standard smart cards from Gemplus, Schlumberger, and/or Infineon can be used for testing smart card readers.

4.1. Simple Windows Logon Configuration Testing - Background

Smart cards directly implement a two-factor authentication policy, and indirectly permit data confidentiality, data integrity, and non-repudiation for multiple applications, including domain logon, secure mail, and secure Web access. Smart cards rely on the public key infrastructure (PKI) of Windows 2000 and Windows XP. Smart cards can only be used to log on to domain accounts, not local accounts. When a password is used to log on interactively to a domain account, Windows 2000 Server and Windows XP Professional use the Kerberos V5 protocol for authentication. If a smart card is used for logon, the operating system uses Kerberos V5 authentication with X.509 v3 certificates unless the domain controller is not running Windows 2000 Server.

To initiate a typical logon session, a user must prove the user's identity to the KDC by providing information known only to the user and the KDC. The secret information is a cryptographic *shared key* derived from the user's password. A shared secret key is symmetric, which means that the same key is used for both encryption and decryption.

To support logging on by using a smart card, Windows 2000 Server implement a public key extension to the Kerberos protocol's initial authentication request. In contrast to shared secret key cryptography, public key cryptography is asymmetric; that is, two different keys are needed — one to encrypt, another to decrypt. Together, the keys needed to perform both operations make up a private/public key pair.

When a smart card is used in place of a password, a private/public key pair stored on the user's smart card is substituted for the shared secret key derived from the user's password. The private key is stored only on the smart card. The public key can be made available to anyone with whom the owner wishes to exchange confidential information.

In the public key extension to the Kerberos protocol, the client encrypts its part of the initial Authentication Service Exchange (AS Exchange) with the private key and passes the certificate to the KDC. The KDC encrypts the user's logon session key with the public half of the user's key pair. The client then decrypts the logon session key by using the private half of the key pair.

Initiating a smart card logon session involves the following process:

1. The user inserts a smart card into a card reader attached to the computer.

2. The insertion of the card signals the SAS just as pressing CTRL+ALT+DEL signals the SAS on computers configured for logging on using a password.

3. In response, Winlogon dispatches to MSGINA, which displays a modified logon dialog box. In this case, however, the user types only the personal identification number (PIN).

4. MS-GINA sends the user's logon information to the LSA just as it does with a logon session using a password.

5. The LSA uses the PIN for access to the smart card, which contains the user's private key along with an X509 v3 certificate that contains the public half of the key pair.

6. The Kerberos SSP on the client computer sends the user's public key certificate to the KDC as pre-authentication data in its initial authentication request.

7. The KDC validates the certificate, extracts the public key, and then uses the public key to encrypt a logon session key. It returns the encrypted logon session key and a TGT to the client.

8. If the client owns the private half of the key pair, it can use the private key to decrypt the logon session key. Both the client and the KDC then use this logon session key in all future communications with one another.

    - All cryptographic operations that use these keys take place on the smart card.

The rest of the authentication process is the same as for a standard logon session.

As the removable security device is inserted in response to the initial prompt for user logon, the system should recognize the “smart card” device being inserted, and then prompt for the user PIN.

Just as with the 802.1X authentication configuration and testing, vendors should ensure that Windows Logon using a smart card functions properly to supplement the user authentication to the domain. Procedures for setting up Windows Logon using a smart card can be found on the Microsoft web site.

4.1.1. Windows Logon Test Process

These tests will be done using a local intranet rather than a corporate network or internet type of connection. The server will be directly connected to the client.

1. Ensure that the user is configured for “Smart Card Logon” within Active Directory on the server.

2. Install the removable security device and software if you have not already done so.

3. Install the “Smart Card User” certificate from the enterprise smart card enrollment station onto the removable security device which can be used for 802.1X authentication. The user name provided will have been registered on the server. This same certificate can be used for 802.1X authentication.

4. Reboot the system. At this point, Windows should provide a different logon prompt which indicates to either “Insert card or press Ctrl-Alt-Delete to begin”.

5. Plug in the removable security device. You should be prompted to enter the access PIN. If Windows is able to successfully validate the credentials stored on the removable security device, you’ll be logged in to Windows.

4.2. Simple 802.1X Authentication Configuration

Windows XP provides native support for IEEE 802.1X (Wired LAN or Wireless LAN) authentication using a smart card. This is a simple intrinsic capability of the operating system that should be tested. The vendor should enable 802.1X authentication to use smart card authentication credentials and ensure that it works properly on a notebook computer.

The following steps can be used to configure a Windows XP notebook for 802.1X authentication using a smart card. The test platform should be preconfigured with 802.11 wireless LAN support and must have a PCMCIA wireless LAN card. Furthermore, a 802.11 network access point (AP) must be available that supports the 802.1X protocol. This AP is connected to a RADIUS server that also includes 802.1X support.

1. Ensure that the system has been restored to its pristine condition using the GHOST image stored on the hard drive of the test system. This particular test will only be done with Windows XP.

2. Install the removable security device and its software support

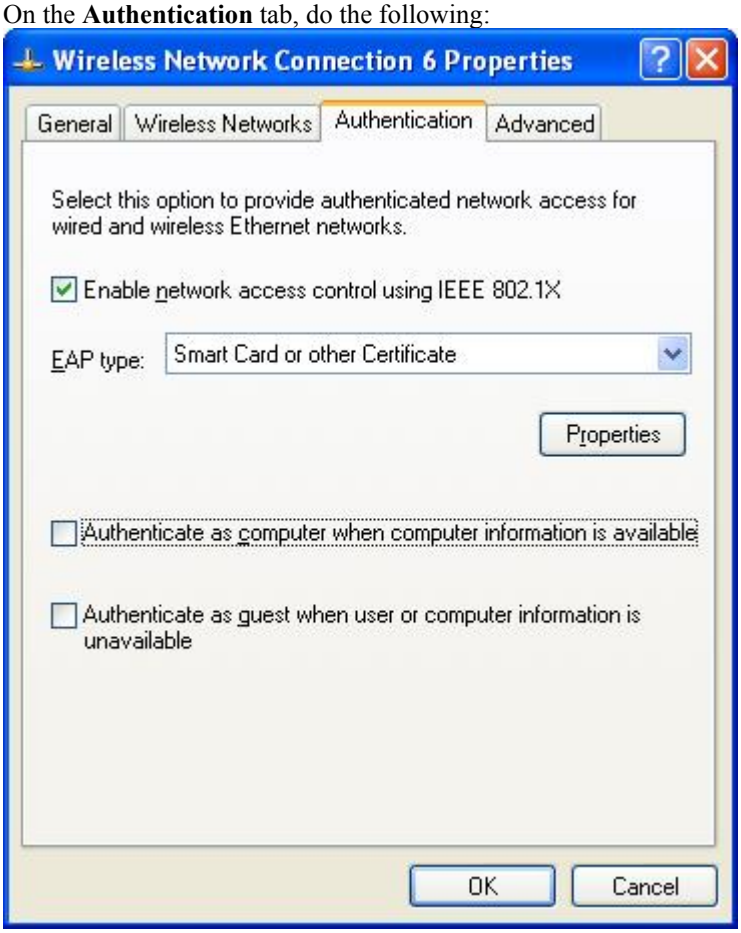
3. The following configuration options require system administrator privileges.

4. Open **Network Connections**

    a. To open Network Connections, click **Start**, click **Control Panel**, click **Network and Internet Connections**, and then click **Network Connections**.

5. Right-click the connection for to be enabled for IEEE 802.1x authentication, and then click **Properties**.

6. On the **Authentication** tab, do the following:



   a. To enable IEEE 802.1x authentication for this connection, select the **Network access control using IEEE 802.1X** check box. This check box is selected by default.

7. In **EAP type**, click the Extensible Authentication Protocol type to be used with this connection.

8. If **Smart Card or other Certificate** in **EAP type** is selected, additional properties can be configured by clicking on **Properties** and, in **Smart Card or other Certificate Properties**, do the following:

   a. To use the certificate that resides on the smart card for authentication, click **Use my smart card**.

   b. To verify that the server certificate presented to your computer is still valid, select the **Validate server certificate** check box, specify whether to connect only if the server resides within a particular domain, and then specify the trusted root certification authority.

   c. To use a different user name when the user name in the smart card or certificate is not the same as the user name in the domain to which the user is logging on, select the **Use a different user name for the connection** check box.

9. To specify whether the computer should attempt authentication to the network if a user is not logged on and/or if the computer or user information is not available, do the following:

a. To specify that the computer attempt authentication to the network if a user is not logged on, select the **Authenticate as computer when computer information is available** check box.

b. To specify that the computer attempt authentication to the network if user information or computer information is not available, select the **Authenticate as guest when user or computer information is unavailable** check box. This check box is selected by default.

- To configure settings on the **Authentication** tab, you must be a member of the local Administrators group.

- For wired and wireless network connections, the settings in the **Authentication** tab apply to the network to currently connected. If the current connection is to a wireless network, the name of the network can be verified by clicking the **Wireless Networks** tab. The name of the network will appear in **Visible networks** and **Preferred networks**, and it will be preceded by an icon with a circle around it.

Once configured for 802.1X authentication using a smart card, connection (either wired or wireless) automatically prompts the user to insert the smart card. Once the smart card insertion is detected, the user is prompted to enter the access PIN to allow the credentials to be retrieved and checked.

For additional information on 802.1X authentication using Windows XP, refer to Microsoft's web site and search for 802.1X.

4.2.1. 802.1X Authentication Test Process

802.1X authentication testing can be done on the same platform for the same removable security device with the same configuration that was used for Windows logon. When testing a different security product, the system should be restored to it's pristine environment using the GHOST image on the system's hard drive.

1. Plug in the wireless LAN card into the PCMCIA slot of the reference system.

2. Windows will detect and configure the card.

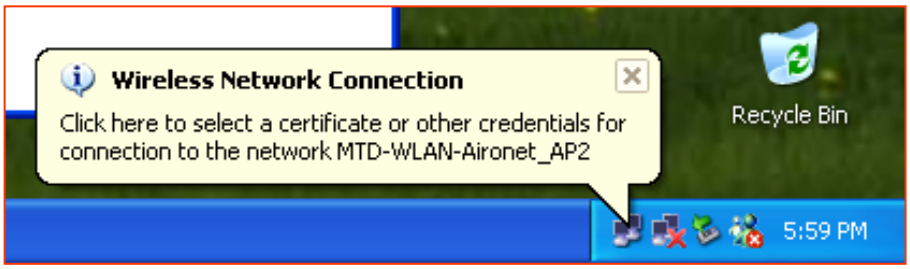
3. Configure the system for 802.1X as described above.

4. Install the removable security device and software if you have not already done so.

5. Install device software support onto the Windows .NET server to allow provisioning of the device with the user certificate.

6. Install the "Smart Card User" certificate from the Windows .NET enterprise smart card enrollment station onto the removable security device which can be used for 802.1X authentication. The user name provided will have been registered on the server. This same certificate can be used for Windows logon.

7. Windows will recognize that the connection requires 802.1X authentication using a certificate found on a "smart card". You will be prompted for the credentials to log onto the wireless LAN as shown in the picture below.



8. Click on that "Wireless Network Connection" balloon to be prompted to insert the security device containing the authentication certificate. You will be prompted to enter the PIN to access the device.

9. Open the Network Connections window (if it is not already open) to view connection status. When the device has been successfully authenticated using the certificate stored on the removable security device, the connection will be given a valid IP address.

4.3. Microsoft Outlook Email: Digitally Signed

NOTE: The process to acquire a new digital ID that each product test will require using Outlook Email is very time consuming and cannot be guaranteed to occur in a given time frame. Verisign's web site says that it can take up to one hour to receive the response.

- Install Removable security device on the platform.

- Get a digital ID for the account to be stored on the device:

    1. With platform connected to the internet with valid Outlook XP email account.

    2. Go to the Verisign web site to download a free digital ID (http://digitalid.verisign.com/cgi-bin/haydn.exe?VHTML_FILE=client/outlook/OESEnrollFree.htm&name=&email=

       You can type that path directly, or go through this process:

        a. Run Microsoft Outlook

        b. Select from menu: Tools >> Options

        c. Open Security tab

        d. Click on "Setup Secure Email" button. In Outlook XP, the button is labeled "Get a Digital ID".

    3. In the Verisign web site, select the "60-day free trial"

    4. The enrollment form includes 4 steps

        a. Enter Digital ID name. Each Digital ID Name has to be unique (you can't use the same one everytime) so you may need to enter a number after each digital ID to make it unique.

        b. Enter Email address

c. Enter Challenge Phrase such as “PortlandBlazers”

d. Select the encryption device from the pull down list. You should see the CSP name for the encryption device being tested.

e. Fill in the rest of the form.

f. Remove the smart card device from the system (remove USB security token or pull the smart card from the reader). This is to make sure that when you finish the form, it actually tries to write to the security device rather than accidentally writing to one of the standard Microsoft software CSPs.

g. Once you hit “Accept” at the bottom of the form, it’ll request a key to be generated from your crypto device.

h. It should ask you to insert the device/card. Once connected, press Enter.

i. It should then prompt you for the PIN to access the device.

j. You then wait for the email response telling you to pick up your digital certificate from Verisign.

5. The email response from Verisign can come relatively quickly (within a few minutes for the experiments that I have done). It includes the path and password to pick up the digital ID. The Verisign instructions do indicate that the email response can take up to an hour to receive. Suggestion: if the email response doesn’t come within 5 or 10 minutes, submit another request for a digital certificate with a new “digital ID name” but using the same email address.

6. Once you have the email from Verisign, it gives directions on how to install the certificate. Follow those instructions to install the certificate to your security device. It should prompt you to enter your access PIN.

7. Add this digital signature to your email as follows:

    a. Within Outlook, go to Tools >> Options >> Security tab

    b. Check the box for “Add digital signature to outgoing messages”

    c. Click on the “Settings” button

    d. “Choose” the certificate from the one you have downloaded to the security device. Do this for both “Signing Certificate” and “Encryption Certificate”. You should see the digital certificate name that was used when you filled out the Verisign request form.

    e. Make sure the box is checked for “Send these certificates with signed messages”

    f. Exit the selection box by pressing OK.

8. Generate an email (send it to your self). As you hit “send”, the security device should prompt for an access PIN so that it can retrieve the digital ID.

9. You will see the email in your Inbox with a red “certificate” icon. Open the email, and open the certificate by clicking on the red icon. Verify the certificate is the one you expected.

4.4. Application Testing Checklist

Table 2: Application Testing Checklist

	Guideline	Functions as Expected Windows 2000	Functions as Expected Windows XP
1.	Digitally signed email can be created using credentials stored and retrieved from the removable security device.	Yes No	Yes No
2.	Device can be easily installed and configured to provide 802.1X authentication credentials	NA	Yes No
3.	Device can be easily installed and configured to provide Windows logon credentials.	NA	Yes No

5. Power Management, Power Consumption, and Performance

Power management and power consumption measurements of the device are to be measured to ensure robust operation of the notebook computer without compromising the battery life of the PC.

5.1. Power Management Testing

The major difference between a notebook computer and a desktop computer lie in the notebook's ability to move from place to place running off battery power. As such, it is important that removable security devices be tested with special emphasis done relative to power management aspects of the computer and interactive effects on the device.

5.1.1. ACPI D-state Testing

Removable security devices should support ACPI D-states as outlined in ACPI 2.0 specifications.

All USB devices must report support D-states D0 and D3 to be USB compliant. It is expected that all these devices not only report D-state support, but also manage transitions to lower power D-states when idle.

5.1.2. ACPI S-state Testing

After installation and configuration of the removable security device, the system should be able to transition into Suspend and then resume without adversely impacting the functionality of the system and/or the device.

1. Ensure that the platform will enter and resume from Standby *before* configuring the removable security device.
    i. Start -> Shutdown -> Standby from the menu.
    ii. System should successfully go into sleep mode.
    iii. Repeat with system running with AC power *and* battery power if possible.
2. Install and configure removable security device
3. Ensure that it is functioning properly by using vendor tools to read configuration data from the device.
4. While the system is running with AC power plugged in, put the system into Suspend
    i. Start -> Shutdown -> Standby from the pulldown menu.
    ii. System should successfully go into sleep mode.
5. Use the system hardware buttons to resume from standby

6. Check functionality of the platform by launching an application from the system menu. This just ensures that an application can be started.

   i. Start -> Programs -> Accessories -> System Tools -> System Information

   ii. Exit the application.

7. Use vendor utility to access the removable security device.

8. Repeat this process at least 3 times, making sure the system is functional after each resume.

9. If system is capable of running using battery power, repeat steps 4-8 with the system using battery power.

5.1.3. ACPI C-state Testing

Basic functional testing of platform ACPI C-states and S-states with the removable security device in the notebook PC. This evaluation to be done only on system running Windows XP.

In order to determine C-state transitions, Windows XP provides a software tool called "PERFMON" which can be configured to display C-state changes.

1. Remove the removable security device from the USB or PCMCIA connection.

2. Click on the Windows "Start" button, and select "RUN"

3. Type PERFMON.EXE and press <Enter>.

4. After starting PERFMON, remove the default configured items.

5. Configure the display (right click on the graph) and add C3-state display to show. It may also be interesting to add C0 and Processor Idle.

6. Simply watch C-state transitions without the USB device plugged in, then plug in the USB device. After the USB device has determine that it is inactive, and has signaled the host that it can be powered down, you should be able to subsequently see the same type of C-state transitions as were seen before plugging the device into USB.

5.2. Power Consumption

Simple power measurements are to be recorded for the following scenarios listed below. Power measurements are to be recording while an application is running to exercise the cryptographic functions of the device. No parameters are defined at this time to identify what is the acceptable power consumption for the device.

Power is to be measured at the connection point of the device (USB or PCMCIA). No other devices should be connected externally to those attach points. Power measurements should be taken for 1 minute or more to generate average and maximum power consumption. The removable security device can perform tests provided by Microsoft's CSPTESTSUITE or alternatively *format / erase* the smart card device or security token.

5.3. Performance

Performance of crypto functions is to be measured by executing Microsoft's CSPTESTSUITE with the "-f" parameter. This provides data for how long it takes a device to perform the test. CSPTESTSUITE reports this data in its results log. This data is to be recorded in Table 4 below.

Performance measurements should be performed with smart card readers using a standard Gemplus, Schlumberger, or Infineon smart card.

5.4. Power Management, Power Consumption, and Performance Checklists

Table 3: ACPI State Verification

Guideline	Functions as Expected Windows 2000	Functions as Expected Windows XP
1. Device and supporting software provides ACPI D-state support to reduce power consumption.	NA	Yes No
2. Device and supporting software does not block ACPI S-state transitions	Yes No	Yes No
3. Device and supporting software allows host processor to enter C3 state as measured in Windows XP using "Perfmon" utility.	NA	Yes No

Table 4: Power and Performance Measurements

Description	Windows 2000	Windows XP
1. **Baseline power reading**: Power measurement should be taken *before connecting* the removable security device. Power should be measured at the attach point (USB or PCMCIA). It is assumed that no power draw should be found.		
2. Device power: The removable security device should be attached and software installed/configured. Once the system is stable (all drivers installed, no more system messages, …), and no applications are running, power should be measured at the attach point for the device (USB or PCMCIA).		

Description	Windows 2000	Windows XP
3. Power measurement with device active: power measurement is taken with device attached/plugged in (smart card inserted into the reader). Start data gathering when application CSPTESTSUITE starts and end power data gathering when test application stops.		
a. Average Power Reading		
b. Peak Power Reading		
4. Performance measurement for all CSPTESTSUITE functions. Smart card used: _____		

6. Low Level Functional Testing

Low level functional testing is to be performed to validate functionality of the device in accordance to industry standard interfaces.

Smart card readers are not expected to run these tests.

6.1. Microsoft CAPI Test Suite

Microsoft Cryptographic API is to be tested for conformance to Microsoft standards using Microsoft's CAPITESTSUITE software test tool.

Steps:

1. Restore system to virgin operating environment using GHOST image stored on separate partition or drive. Evaluation is to be run for both Windows 2000 and Windows XP.
2. Install hardware device and support software (device driver and CAPI CSP)
3. Ensure that the device has a CSP registered with the system by running Microsoft's CAPITESTSUITE within a DOS box without any command line parameters.
4. Use CAPITESTSUITE to test all CSP functions by running the tool
    a. Ensure that CSP is signed.
    b. Functional testing – test all CAPI interfaces using CAPITESTSUITE

6.2. Low Level Functional Test Checklist

Table 5: Low Level CAPI Testing Checklist

	Function	Functions as Expected Windows 2000	Functions as Expected Windows XP
1.	0: CryptAcquireContext	Yes No	Yes No
2.	1: CryptGetProvParam	Yes No	Yes No
3.	2: CryptSetProvParam	Yes No	Yes No
4.	3: CryptReleaseContext	Yes No	Yes No
5.	4: CryptGenRandom	Yes No	Yes No

	Function	Functions as Expected Windows 2000	Functions as Expected Windows XP
6.	5: CryptCreateHash	Yes No	Yes No
7.	6: CryptDestroyHash	Yes No	Yes No
8.	7: CryptGetHashParam	Yes No	Yes No
9.	8: CryptSetHashParam	Yes No	Yes No
10.	9: CryptHashData	Yes No	Yes No
11.	10: CryptGenKey	Yes No	Yes No
12.	11: CryptDeriveKey	Yes No	Yes No
13.	12: CryptDestroyKey	Yes No	Yes No
14.	13: CryptGetKeyParam	Yes No	Yes No
15.	14: CryptSetKeyParam	Yes No	Yes No
16.	15: CryptEncrypt	Yes No	Yes No
17.	16: CryptDecrypt	Yes No	Yes No
18.	17: CryptHashSessionKey	Yes No	Yes No
19.	18: CryptExportKey	Yes No	Yes No
20.	19: CryptImportKey	Yes No	Yes No
21.	20: CryptGetUserKey	Yes No	Yes No
22.	21: CryptSignHash	Yes No	Yes No
23.	22: CryptVerifySignature	Yes No	Yes No